Lab #5 Report: Monte Carlo Localization of a Model Racecar Using LiDAR

Team #5

Kelsey Fontenot
Yifan Kang
Dante Suazo
He'yun Zhi

6.4200 Robotics: Science & Systems

April 10, 2026

1 Introduction (Yifan)

At the core of any autonomous system lies a fundamental question: "Where am I?" Localization, the process by which a vehicle estimates its precise position and orientation relative to a known environment, answers this question. Without a robust localization stack, higher-level capabilities such as path planning and trajectory optimization have no reliable state to operate on. This dependency is especially acute in autonomous racing, where vehicles operate at the limits of their control margins and small positional errors compound rapidly into unsafe behavior.

In this lab, we designed and implemented a complete localization system for the MIT RSS racecar platform, serving as the foundational module upon which future labs — including path planning and obstacle avoidance — will build. The system must estimate the car's 6-DOF pose in real time at greater than 20 Hz, robust to sensor noise and wheel slip.

Our technical solution is Monte Carlo Localization (MCL), also known as a particle filter. We represent the robot's belief over its pose as a weighted set of N hypotheses (particles). At each timestep, two models update the belief: a motion model propagates particles forward using noisy wheel odometry from the VESC, and a sensor model reweights them by comparing the onboard LiDAR scan against ray-traced predictions through a known occupancy map. Particles inconsistent with the observed scan are down-weighted and eliminated through

resampling; those that match are duplicated. Over successive updates, the distribution collapses around the true pose.

We validated the system in both simulation and on the physical car in the Stata basement, demonstrating that our particle filter is both stable in simulation and in real-world.

2 Technical Approach (Dante)

2.1 Motion Model

The motion model propagates each particle forward in time using the twist component of the VESC wheel odometry. At each odometry callback, we extract the body-frame linear velocities v_x, v_y and the angular velocity ω, then integrate over the elapsed interval Δt to obtain a displacement $(dx, dy, d\theta) = (v_x \Delta t,\ v_y \Delta t,\ \omega \Delta t)$. Because each particle maintains its own heading θ_i, we rotate the body-frame displacement into the world frame using a 2D rotation:

$$\begin{bmatrix} dx_w \\ dy_w \end{bmatrix} = \begin{bmatrix} \cos\theta_i & -\sin\theta_i \\ \sin\theta_i & \cos\theta_i \end{bmatrix} \begin{bmatrix} dx \\ dy \end{bmatrix} \tag{1}$$

To model the uncertainty inherent in wheel odometry — tire slip, encoder quantization, mechanical backlash — we add zero-mean Gaussian noise whose magnitude scales with the amount of motion:

$$\sigma_{xy} = 0.05\, d + 0.01, \qquad \sigma_\theta = 0.05\, |d\theta| + 0.01 \tag{2}$$

where $d = \sqrt{dx^2 + dy^2}$ is the translational displacement. The proportional terms (0.05) capture the fact that faster driving yields less reliable odometry, while the constant drift terms (0.01) ensure particles continue to spread during near-stationary periods, preventing premature collapse.

2.2 Sensor Model

The sensor model evaluates how well each particle's hypothesized pose explains the observed LiDAR scan. For each particle, we ray-trace through the known occupancy map to compute expected ranges for 99 evenly-spaced beams, downsampled from the raw 1080 using an angle stride of 11. Each expected range d is compared against the corresponding measured range z using a precomputed probability table.

The probability $p(z \mid d)$ is a weighted mixture of four distributions, each modeling a distinct physical phenomenon:

$$p(z \mid d) = \alpha_{\text{hit}}\, p_{\text{hit}} + \alpha_{\text{short}}\, p_{\text{short}} + \alpha_{\text{max}}\, p_{\text{max}} + \alpha_{\text{rand}}\, p_{\text{rand}} \tag{3}$$

- p_{hit}: A Gaussian centered at the expected range d with $\sigma = 8$ (in discretized index units), representing accurate measurements ($\alpha = 0.74$).
- p_{short}: A linearly decaying distribution for unexpected obstacles closer than the map predicts ($\alpha = 0.07$).
- p_{max}: A spike at the sensor's maximum range for beams that miss all surfaces ($\alpha = 0.07$).
- p_{rand}: A uniform distribution over all ranges, modeling random sensor noise ($\alpha = 0.12$).

These weights heavily favor accurate measurements while providing graceful degradation for real-world phenomena such as unmapped obstacles, specular reflections, and sensor saturation. Rather than evaluating this mixture at runtime for every beam of every particle, we precompute a 201×201 lookup table indexed by discretized (z, d) pairs at startup. This reduces the per-beam cost from four distribution evaluations to a single array access — critical for real-time performance on the Jetson.

The total particle weight is the product of per-beam probabilities across all 99 beams, computed in log space to prevent numerical underflow. We further apply a squashing exponent of $\frac{1}{3}$, which broadens the weight distribution and mitigates particle starvation — a failure mode where one dominant particle captures all resampling probability.

2.3 Particle Filter

The particle filter integrates the motion and sensor models into a Bayesian estimation loop operating on $N = 100$ particles, each representing a pose hypothesis (x, y, θ) in the map frame.

Initialization. When the operator publishes a pose estimate via RViz, 100 particles are drawn from a Gaussian centered at the selected location with $\sigma_{xy} = 0.5\,\text{m}$ and $\sigma_\theta = 0.15\,\text{rad}$. This spread accommodates the imprecision of a manual click while remaining narrow enough for rapid convergence.

Prediction. Each odometry message triggers the motion model (Equation 1), which propagates all particles forward with motion-scaled noise (Equation 2). This step runs at approximately 50 Hz, matching the VESC odometry rate.

Update and resampling. Each LiDAR scan triggers the sensor model, which assigns a weight to every particle via the precomputed table. We then perform multinomial resampling: particles are redrawn with replacement proportional to their weights. High-weight particles are duplicated; inconsistent ones are eliminated.

After resampling, we inject post-resample Gaussian noise ($\sigma_{xy} = 0.2\,\text{m}$, $\sigma_\theta = 0.05\,\text{rad}$) to prevent particle collapse — a well-known failure mode where all particles converge to identical copies, eliminating the diversity needed to track future motion. This noise level represents a deliberate tradeoff: too little and the filter loses diversity; too much and convergence degrades.

Pose estimation. The output pose is the arithmetic mean of particle positions and the *circular mean* of headings:

$$\hat{\theta} = \text{atan2}\left(\frac{1}{N}\sum_i \sin\theta_i,\ \frac{1}{N}\sum_i \cos\theta_i\right) \tag{4}$$

This avoids the discontinuity at $\pm\pi$ that would corrupt a naive angle average. The estimated pose is published as an Odometry message and a TF transform linking the `map` frame to `base_link`.

Table 1 summarizes the key parameter values used in our final system.

Table 1: Key particle filter parameters and their values.

Parameter	**Value**	**Rationale**
Particles (N)	100	Sufficient coverage while maintaining >20 Hz updates
LiDAR beams	99	Every 11th beam from 1080; 11× less computation
α_{hit}	0.74	Dominant weight on accurate measurements
σ_{hit}	8.0	Tolerance in discretized index units
Weight squash	$w^{1/3}$	Prevents single-particle dominance
Post-resample σ_{xy}	0.2 m	Balances diversity against convergence
Post-resample σ_θ	0.05 rad	$\approx 3°$; prevents heading collapse

3 Quantitative Results

3.1 Localization Performance on the Real Car (Yifan)

To evaluate the particle filter's localization accuracy, we compared its pose estimate against raw VESC wheel odometry over a 75-second driving run in the Stata basement. The particle filter estimate was extracted from the `map` → `base_link` transform broadcast to `/tf`, while VESC odometry was taken from the `odom` → `base_link` transform. Since these two transforms live in different reference frames — the PF operates in the global map frame while the VESC

integrates displacement in its local odom frame — a naive subtraction would conflate frame misalignment with genuine drift. We therefore applied a continuous heading correction: at each timestep, the incremental VESC displacement was rotated by the instantaneous heading difference between the PF and VESC yaw estimates before being accumulated. This isolates the purely translational component of odometry error, independent of frame initialization artifacts.

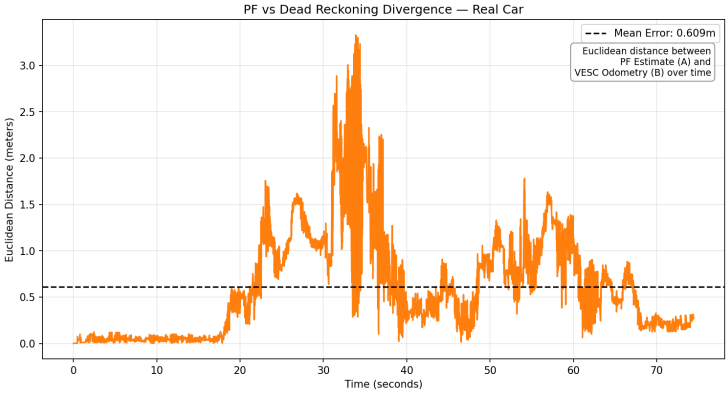

Figure 1: Divergence between the particle filter trajectory and heading-corrected VESC odometry over the 75-second run.

The resulting divergence between the PF trajectory and heading-corrected VESC odometry is shown in Figure 1. The mean positional error is 0.609m, with a peak of approximately 3m occurring around $t = 30$s, coinciding with a sharp turn where wheel slip is most pronounced. Outside of dynamic maneuvers, the error remains consistently below 1m, demonstrating that the particle filter successfully anchors the pose estimate against LiDAR-observed map features even as the underlying odometry drifts.

The trajectory comparison (Figure 2) further illustrates this: the VESC odometry path (blue) diverges visibly from the PF path (red) at the bottom cluster where the car reverses direction, and again near the top of the run. The PF trajectory, by contrast, traces a geometrically consistent path that aligns with the known map structure.

3.2 Limitations of VESC Odometry as Ground Truth (Yifan)

While this comparison effectively demonstrates that the particle filter outperforms dead reckoning, using VESC odometry as the baseline carries an important caveat: it is not true ground truth. The VESC measures wheel rotation,

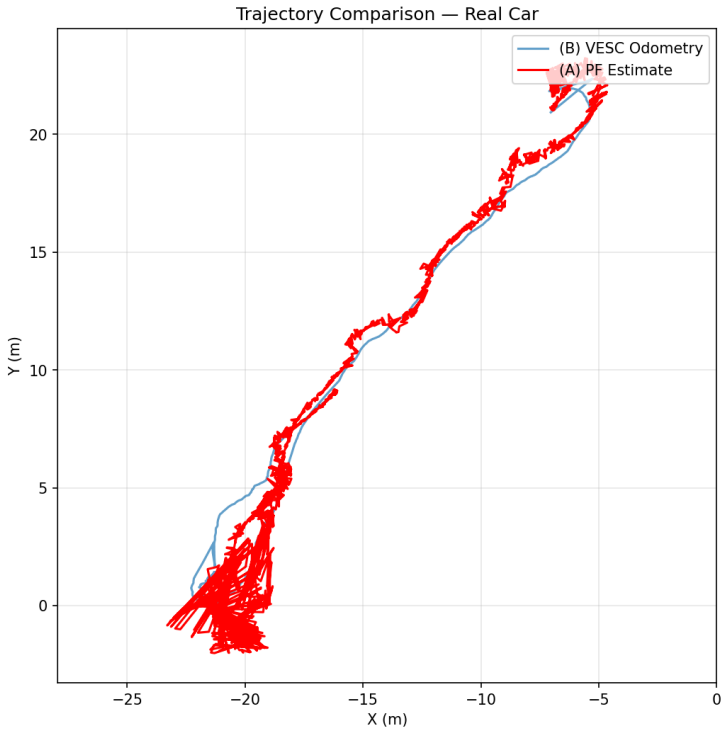

Figure 2: Trajectory comparison showing the VESC odometry path (blue) diverging from the PF path (red) during direction changes.

which introduces several sources of systematic error — tire slip on smooth floors, encoder quantization, and mechanical backlash — that accumulate monotonically over time. As a result, the divergence metric reported here measures how much the PF differs from dead reckoning, not how close the PF is to the car's true physical position.

A rigorous ground truth would require an external reference system such as a motion capture array or a high-precision RTK-GPS, neither of which was available in this setting. In the absence of such a system, the particle filter's self-consistency — the degree to which the laser scan aligns with the known map at the estimated pose — serves as a proxy for accuracy. Qualitatively, we observed strong scan-to-map alignment throughout the run, suggesting the PF estimate is reliable even though it cannot be independently verified against a true ground truth.

3.3 Simulation vs Real (Kelsey)

Simulation testing showed off a well working particle filter with large spikes correlating to turns. We found that our particle filter had a low mean error of 0.308m. In real life, we observed the mean error to be doubled as in Figure 2 and looked at other metrics to narrow down the cause. The number of effective particles over the course of the 75-second run is shown in Figure 3 and shows that the particle filter remains stable. However, several sharp drops that indicate temporary localization failures. These failures, particularly at the 32 second mark are in line with the large spike in mean error in Figure 2 at the same time.

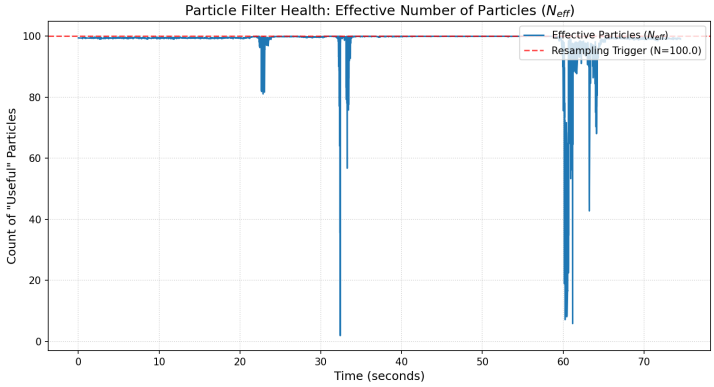

Figure 3: Stable particle filter with some drops that correspond to localization failures.

To investigate this further, we looked at sensor model innovation over time.

Innovation refers to the difference between what the sensors are reading as compared what our particle filter is predicting. Low innovation means the model is doing well, and high innovation means there is a mismatch between the prediction and the sensors, in this case our LIDAR readings. Figure 4 reveals that at the 32 second mark, innovation spikes from low to high, which matches with previous plots. During the run, this correlated when the robot was being driving in fast circles in an area of the map that did not directly match real life. There were several large boxes and bikes that the LIDAR picked up that weren't on the provided map that the model was predicting. In combination with fast erratic driving, the model struggled with localization in this area. The particle filter eventually recovered when it drove back into the hallway around the 60 second mark and the improvement can be seen at that mark on the rest of the plots.

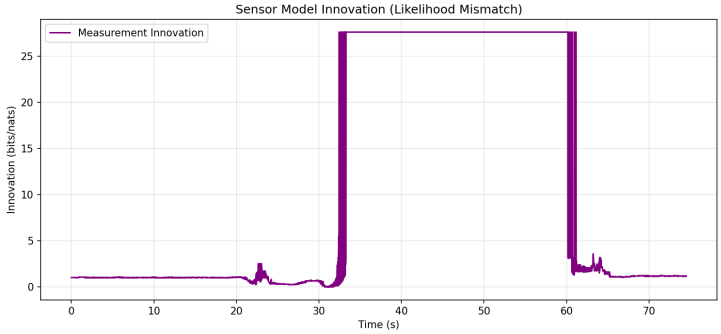

Figure 4: Model prediction mismatch over course of run gives indication as to why mean error has large spikes.

3.4 Runtime Performance (Dante)

Our particle filter achieved a sustained update rate of approximately 71 Hz on the Jetson Orin with 100 particles and 99 LiDAR beams, corresponding to a per-update latency of roughly 14 ms. This exceeds the 20 Hz real-time requirement by a factor of 3.5×. The dominant computational cost is the sensor model's ray-tracing step, which scales linearly with both particle count and beam count. The precomputed 201×201 lookup table eliminates per-beam distribution computation, reducing the sensor model to array indexing and a single log-sum operation per particle. We did not conduct a systematic sweep of particle count versus update rate due to time constraints; however, the 3.5× headroom suggests the system could support approximately 350 particles before dropping below the real-time threshold, assuming linear scaling.

4 Media

Please see our localization model visualized in gif form here and access the raw videos here.

5 Conclusion (Kelsey)

In this lab, we developed and implemented a localization system for our racecar using a particle filter. Our particle filter used a combined motion model and sensor model to estimate the racecar's position on the map.

In real life, our results showed that our particle filter performed best within the hallway outside the classroom because that section was most similar to the map. If there were large objects such as bikes or boxes in real life that did not appear in the map, our particle filter would struggle because it was not expecting those objects there.

Overall, our lab showed success in our particle filter both in simulation and real life, though there are still limitations to its performance in real life especially with erratic behavior. Moving forward, we may want to look into the quantitatively defining the limitations of our localization method (ie. car velocity and turn angle as it correlates to error) to inform how we design our path planning and path following algorithms.

6 Lessons Learned

6.1 Kelsey

One of the biggest takeaways from this lab was the importance of understanding the data and knowing if it is being plotted incorrectly. At first, I had trouble with some of the data visualization because the odometry error between the particle filter and ground truth was extremely off and this didn't match the observed behavior. Specifically, there was an issue with aligning the data due to rotations and translation in the XY plane. Once this was resolved, the correct graphs were plotted, thus it is important to understand how expected behavior might appear in data visualization.

6.2 Yifan

This localization lab underscored that successful autonomous vehicle deployment relies just as much on team coordination and robust infrastructure as it does on accurate algorithms. Because tuning a localization stack requires constant adjustments to parameters and ROS nodes, keeping everyone strictly on the same page regarding code versioning and configurations was crucial to avoid

compounding errors in the car's estimated pose. Furthermore, I realized the importance of a stable internet connection. Since we relied on the network to SSH into the vehicle and visualize LiDAR data, intermittent Wi-Fi drops severely disrupted our testing and debugging process.

6.3 Dante

A major takeaway from this lab was that infrastructure decisions can consume as much debugging time as the algorithm itself. Docker volume mounts, TF frame naming, and RViz QoS settings each produced subtle failures that were difficult to diagnose remotely over an unstable WiFi link to the car. Documenting these operational details turned out to be just as important as documenting the code, because without a clear record of what worked, every reconnection risked repeating the same mistakes.

6.4 He'yun

I learned that it is important for everyone to confidently understand the high-level fundamentals of a program before debugging, testing, and evaluating. Charging into a project without taking the time to understand its overall structure and motivations, as well as the workings of the hardware itself, leads to hindered communication and compounding errors.